Oskari Peltonen

HTML 5 JA SEN UUDET RAJAPINNAT

Tietojenkäsittelyn koulutusohjelma

2013

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

HTML 5 JA SEN UUDET RAJAPINNAT

Peltonen, Oskari

Satakunnan ammattikorkeakoulu

Tietojenkäsittelyn koulutusohjelma

Toukokuu 2013

Ohjaaja: Grönholm, Jukka

Sivumäärä: 34

Liitteitä: 4

Asiasanat: HTML5, Internet-kehitys, Ohjelmointirajapinnat

Opinnäytetyön aiheena oli tutkia uutta HTML 5 spesifikaatiota ja mitä uutta se tuo web-kehitykselle, painottuen uusille ohjelmointirajapinnoille. Aihetta tutkittiin sekä web-kehittäjän että loppukäyttäjän näkökulmasta, sekä käytiin läpi spesifikaation kehitysvaiheet.

HTML 5 AND IT'S APPLICATION PROGRAMMING INTERFACES

Peltonen, Oskari

Satakunta University of Applied Sciences

Degree Programme in Information Technologies

May 2013

Supervisor: Grönholm, Jukka

Number of pages: 34

Appendices: 4

Keywords: HTML5, web development, programming interfaces

The purpose of this thesis was to research the new HTML 5 specification and what new it brings to web-development, emphasizing on new programming interfaces. The study was carried out from both web-developer and end user's perspective, and the stages of development of the specification were gone through.

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön aihe on HTML 5 ja sen uudet rajapinnat. Selvitän uuden HTML spesifikaation merkittävämpiä muutoksia, uusien ominaisuuksien toimintalogiikkaa sekä näiden merkitystä web-kehityksessä. Koska tuleva standardi on niin laaja kokonaisuus niin rajaan aiheen loppukäyttäjien kannalta olennaisimpiin muutoksiin sekä uudistuksiin.

Ohjelmointirajapinnat ovat erilaisia käyttöliittymiä, joiden avulla saadaan eri ohjelmat ”keskustelemaan” keskenään. HTML 5 tarjoaa lukuisia uusia rajapintoja, joiden avulla web-sivuista saadaan dynaamisempia ja monipuolisempia. Monet selainlaajennokset joiden avulla on totuttu näyttämään sisältöä web-sivuilla, voivat tulla tarpeettomiksi koska uudessa spesifikaatiossa on monia ominaisuuksia web-sivujen luomiseen. Näiden avulla web-sivut toimivat ilman erillisiä tukiohjelmia, esimerkiksi Adobe Flash player animaatioon sekä videon suoratoistoon.

Mobiililaitteilla HTML 5 on erityinen edistysaskel, koska sen avulla voidaan toteuttaa palveluita web-sovelluksina perinteisten natiivi-sovellusten sijaan. Etenkin sovellusten kehittäjille tämä hyvä asia, koska kehityskustannukset vähenevät kun ei tarvitse toteuttaa sovellusta monelle eri alustalle. Mobiilikäyttäjien ei tarvitse asentaa web-sovellusta laitteelleen, sekä päivitykset tapahtuvat palvelinpäässä joten niistäkään ei käyttäjän tarvitse huolehtia. Tulevassa standardissa on monia uusia ominaisuuksia, jotka voivat tehdä perinteiset selainlaajennokset tarpeettomiksi.

HTML 5 –standardin kehitys on ollut pitkä prosessi. Sen kehitys aloitettiin vuonna 2004, ja sen lopullinen versio julkaistaneen vuonna 2014. Uusi standardi on nykyisin jo laajassa käytössä, etenkin mobiili-sovelluksissa sekä videon suoratoistopalveluissa.

2 HTML 5:N KEHITYSHISTORIA

2.1 HTML yleisesti

Vuonna 1989 Tim Berners-Lee kehitti World Wide Webin työskennellessään Cernissä, Sveitsissä. Sen tavoitteeena oli helpottaa tietojen kulkua tutkijoiden välillä niin Cernissä kuin ympäri maailmaa. Berners-Lee halusi kehittää World Wide Webistä ihmisille hyödyllisen. Lokakuussa 1990 hän oli määritellyt kolme keskeistä teknologiaa, jotka ovat www:n perusta: HTML, Url ja HTTP. (World Wide Web Foundation 2008.) HTML on lyhenne sanoista Hypertext Markup Language eli hypertekstin merkintäkieli. Sitä käytetään kuvaamaan hypertekstiä www-verkossa. HTML:n avulla määritellään web-sivujen rakennetta ja asettelua. (Bellis 2013.)

HTML:n ensimmäinen versio on julkaistu vuonna 1991. Siinä oli 22 erilaista elementtityyppiä, joista on yhä käytössä kolmetoista (Berners-Lee 1991.). HTML:n kieli oli aluksi hyvin rajoittunutta, eikä sillä voinut tehdä kuin yksinkertaisia tekstejä World Wide Web:iin (Shannon 2012). HTML suunniteltiin aluksi SGML:n (Standard Generalized Markup Language) sovellukseksi, mutta se toteutettiin vasta vuonna 1993. Kyseinen määrittely raukesi jo kuuden kuukauden päästä, kun Dave Ragget julkaisi kilpailevan HTML+:n, joka sisälsi jo käytössä olevat lomakekentät ja taulukot. (Berners-Lee 1991.)

HTML 2.0 kehitettiin vuonna 1994 Internet Engineering Task Forcen eli IETF:n toimesta (Boumphrey 2001, 9). Se sisälsi kaikki HTML 1.0:n alkuperäisestä määrittelystä ja siihen oli lisätty muutamia uusia ominaisuuksia mukaan. Vuoteen 1997 saakka HTML 2.0 oli standardi websivujen kehitykselle ja se määritti monia

uusia HTML elementtejä. (Shannon 2012.) HTML 2.0 standardilla oli merkittävä osa World Wide Webin suosion kasvussa (Wilson 2013).

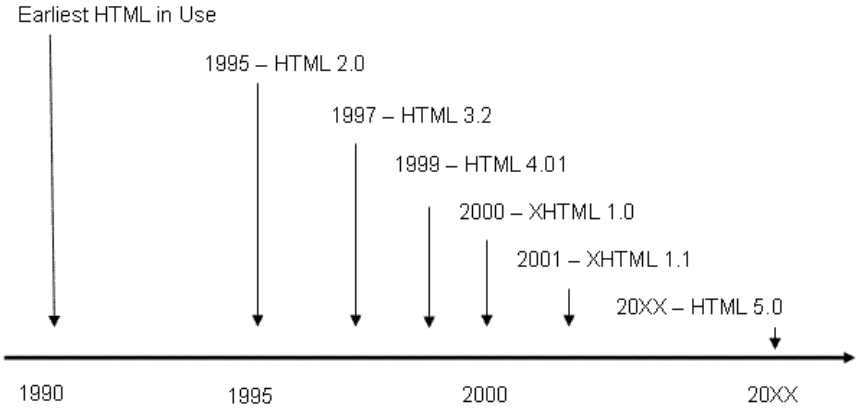
HTML 3 sisälsi monia uusia paranneltuja ominaisuuksia ja keinoja, joilla web-sivujen kehittäjät pystyivät entistä paremmin päättämään, miten he esittävät tiedon sivuillaan. Selainten kehittäjät olivat hitaita toteuttamaan uusia ominaisuuksia ja vain muutamia ominaisuuksia lisättiin, joten osittain siksi HTML 3.0 oli nopeasti unohdettu. (Shannon 2012.) HTML 3.2 kehitettiin alkuvuodesta 1996 Worl Wide Web Consortiumin (W3C) toimesta (Boumphrey 2001, 9).

HTML 4.0 oli suuri edistysaskel HTML-standardille, ja viimeinen kehitysvaihe klassiselle HTML:lle. Vanhasta HTML 3:sta tuotiin ominaisuuksia, ja vanhoja tageja muokattiin. Uuteen HTML –versioon kehitettiin myös uusi edistyksellienn kieli, CSS eli Cascading Style Sheets. HTML 4.0:sta tuli virallinen standardi huhtikuussa 1998, W3C:n toimesta. Selaintuki tuli yllättävän helposti Microsoftin Internet Explorer –tuotteilta, joilla oli hyvä tuki uuden standardin tageille ja attribuuteille. Kun HTML 4.0 oli ollut käytössä jo jonkin aikaa, käytiin dokumentaatio läpi uudelleen ja siihen tehtiin pieniä muutoksia. Versio nimettiin HTML 4.01:ksi, ja se on viimeinen versio kyseiselle standardille. (Shannon 2012.)

2000-luvun alussa julkaistiin XHTML 1.0 (Shannon 2012). XHTML on lyhenne sanoista Extensible Hypertext Markup Language. Se on muuten sama kuin HTML 4.01 paitsi, että se kirjoitettiin uudelleen XML:n (Extensible Markup Language) sääntöjen mukaisesti ja siinä on mukana XML:n täsmällisyyksiä; koodin on oltava täydellistä kun se menee lukijan selaimeen (Boumphrey 2001, 7, 9., Shannon 2012). XHTML:n käyttö mahdollisti uusia toimintoja sekä käyttömahdollisuuksia (Shannon 2012).

HTML:n kehitys on pitkälti ollut selaimiin jo tehtyjen laajennusten uudelleenkirjaamista ja tarkkaa määrittelyä. Korpelan (2011, 24) mukaan sellaiset uudet piirteet, jotka on määritelty muilla tavoin, ovat jääneet toteuttamatta tai ne on toteutettu hitaasti tai epäyhtenäisesti. HTML 5 on saanut alkunsa ajatuksesta, että HTML:ää ja selaimia kehitetään rinnakkain ja selainten valmistajat toimivat yhteistyössä. Kuvassa 1. havainnollistetaan standardin kehitys vuodesta 1990

nykypäivään asti. HTML 5:en standardointivuotta on esitetty niinkin pitkälle kuin vuoteen 2022, mutta näillä näkymin lopullinen versio ilmestyy vuonna 2014. (W3C 2011.)



Kuva 1. HTML:n kehitys.

2.2 Kehitysryhmät

The Web Hypertext Application Technology Working Group eli WHATWG on kasvava yhteisö ihmisiä, jotka ovat kiinnostuneita World Wide Webin kehityksestä. WHATWG keskittyy pääasiassa HTML:n kehitykseen sekä rajapintoihin, joita tarvitaan websovelluksissa. WHATWG:n perustajissa on yksilöitä Applelta, Mozillalta ja Operalta. Se on perustettu vuonna 2004, jolloin vaikutti ettei W3C:llä ole enää kiinnostusta kehittää HTML:ää. (WHATWG 2013.)

WHATWG:n painopiste on HTML standardissa sekä erilaisissa ohjelmointirajapinnoissa. Aiemmin The Web Hypertext Application Technology Working Group on työstänyt esimerkiksi Web Forms 2.0 ja Web Controls 1.0. Web Forms 2.0 on integroitu HTML5 standardiin. (WHATWG 2013.)

W3C eli World Wide Web Consortium on perustettu vuonna 1994 ohjaamaan World Wide Webin kehittymistä täyteen mittaansa. Perustaja oli sen nykyinenkin johtaja Tim Berners-Lee. (W3C 2012.) W3C kehittää yhteisiä ja yhteensopivia World Wide

Webin pelisääntöjä ja teknologioita kuten spesifikaatioita, ohjeita, ohjelmistoja sekä työkaluja (Nykänen 2003).

W3C:llä on ympäri maailmaa noin 400 jäsenorganisaatiota. Sen toiminnasta hallinnollisesti vastaa kolme tutkimusinstituuttia Pohjois-Amerikassa, Euroopassa sekä Aasiassa. Noin seitsemänkymmennen henkilön vahvuinen W3C-tiimi vastaa W3C:n jatkuvasta toiminnasta. (Nykänen 2003.)

W3C:n suosituksista suurin osa on teknisiä eli ne ohjeistavat jonkin välineen toteuttamista tai käyttöä. Tyypillisiä välineitä ovat esimerkiksi WWW-selain tai tekstinkäsittelyohjelma tai jokin näiden osa. Suositusten noudattaminen takaa sen, että eri toimijoiden työ on yhteensopivaa, esimerkiksi Web-sivut näkyvät oikein eri selaimissa jo dokumenttien koneellinen lukeminen onnistuu ongelmitta. Suositusten lisäksi W3C:n työryhmät tuottavat myös ohjeita sekä erilaisia ohjelmistoja. Ne kirjaavat asiantuntijayhteisön tietoa käyttökelpoiseen muotoon ja tarjoavat perustan testata ja käyttää uudenlaisia teknologioita. (Nykänen 2003.)

2.3 HTML 5 luonnoksesta standardiksi

WHATWG-ryhmä alkoi työstää uutta standardia vuonna 2004, kun W3C päätti ettei julkaise enää uutta versiota XHTML:stä. Nykyisin W3C ja WHATWG työskentelevät yhdessä HTML 5:n kehityksessä. (W3C 2009.) Niiden suhdetta voisi kuvailla sellaiseksi, että WHATWG:ssa toteutetaan kehitystyö ja W3C hyväksyy sen tuloksia ja myöhemmin vahvistaa ne standardiksi eli W3C:n suositukseksi (Korpela 2011, 58).

HTML 5:n kehitystyö on ollut hidasta, sen voidaan sanoa alkaneen vuonna 2004 ja saaneen virallisen aseman vuonna 2007, mutta sen ei oleteta valmistuvan vielä moneen vuoteen. Mutta toisaalta sen kehitys on alkanut kulkea selainten kehityksen kanssa rinnakkain, eli sen piirteitä on jo toteutettu melko laajasti. (Korpela 2011, 58.) Esimerkiksi pienemmissä ja tarkemmalle käyttäjäryhmälle suunnatuissa web-sovelluksissa joitakin ominaisuuksia voi jo käyttää. Lisäksi internetistä löytyy useita

sivustoja, jotka pitävät listaa eri selainten ja selainversioiden tukemista HTML 5:n ominaisuuksista. (Jaakkola 2011.)

World Wide Web Consortium eli W3C järjesti vuonna 2004 kokouksen, jossa päätettiin aloittaa HTML:n seuraavan version kehittäminen uudelta pohjalta. Eri mieltä olleet henkilöt muodostivat WHATWG:n eli Web Hybertext Applications Technology Working Croup:n, joka alkoi kehittämään laajennettua lomakekäsitettä eli Web Forms 2.0 sekä muita laajennuksia. WHATWG julkisti Web Applications 1.0 –nimisen luonnoksen HTML:n laajentamisesta vuonna 2005 ja vuotta myöhemmin W3C:n johtaja Tim Berners-Lee totesi WHATWG:n saavuttaneen työllään menestystä ja ilmoitti W3C:n tulevan tekemään yhteistyötä WHATWG:n kanssa. (Korpela 2011, 58.)

W3C perusti HTML-työryhmän uudelleen vuonna 2007. Web Applications 1.0 korvattiin nimellä HTML 5. Aluksi sen rinnalla toimi toiselta pohjalta toiminut XHTML-työryhmä, mutta se lakkautettiin kuitenkin vuoden 2009 lopussa. W3C julkisti ensimmäisen HTML 5-luonnoksen vuonna 2008. (Korpela 2011, 58.)

HTML 5:n kehittäminen jatkuu sekä W3C:ssä että WHATWG:ssä. Työnjakoa voisi kuvata sellaiseksi, että WHATWG tekee kehitystyön ja W3C hyväksyy sen tuloksia ja aikanaan vahvistaa ne standardiksi eli suositukseksi. Kehitystä on vaikea seurata, koska molempien tahojen luonnokset ovat osittain rakenteeltaankin erilaisia, joten niiden vertailu on vaikeaaa. Kuitenkin molemmilla tahoilla editorina toimii Ian ”Hixie” Hickson (Google Inc.), joka generoi määrittelyt eri asetuksia käyttäen samasta tietokannasta. (Korpela 2011, 58.)

Määrittelyt sisältävät paljon erikoistermejä ja yksityiskohtaisia teknisiä määrittelyjä. W3C:n määrittelyt ovat helppolukuisempia, ja niiden sisältö on W3C:n työryhmän käsittelyn läpäissyttä. Tämä tarkoittaa sitä, että se sisältö on todennäköisemmin sellaista, joka voi päästä lopulliseen määrittelyyn ja toteutuksiin. (Korpela 2011, 58.)

W3C on ilmoittanut uudesta suunnitelmasta, jonka mukaan HTML 5 –spesifikaatiot siirretään vuoden 2014 loppuun mennessä ”Recommended” –tilaan. HTML 5.1 –spesifikaatiot siirretään ”Recommended” –tilaan vuoden 2016 loppuun mennessä.

Tila merkitsee W3C:n termeillä valmista, lopullista versiota standardista. Aiempien suunnitelmien mukaan HTML 5:n piti valmistua vasta vuonna 2022, mutta W3C on todennut, että pienemmät ja nopeammat päivitykset standardeihin ovat parempi vaihtoehto pitkän kehityskaaren sijasta. (Bright 2012.)

2.4 Ratkaisu webin ongelmiin

Nykypäivänä web-sivuja voidaan selata monella erilaisella laitteella. Siksi web-kehittäjien täytyy ottaa huomioon monet saatavuusongelmat käyttöjärjestelmien ja laitteiden kokoonpanoissa. Esimerkiksi Applen monet tuotteet rajoittavat Flashin käyttöä, sekä videon toistossa erilaiset soittimet ja codecit vaihtelevat paljon. Tälläiset yhteensopivuusongelmat huonontavat käyttökokemusta sekä luovat kuilun eri laitteiden välillä. (Cardinal 2011.)

HTML 5:ttä alettiin suunnittelemaan, jotta saataisi ratkaistuksi erilaiset yhteensopivuusongelmat sekä tekemään web-sivut käyttäjäystävällisemmiksi mobiililaitteilla. Koska webin selaaminen on räjähdysmäisesti kasvanut mobiililaitteilla, uudessa standardissa on mm. erilaisia metatageja, joilla parannetaan koko-näytöllä selaamista sekä toimivampaa zoomausta. (Cox 2011.)

Uudessa standardissa videon toistoa on pyritty sisällyttämään selaimeen natiivisti, eli ilman erillisiä selainlaajennuksia tai hankaloittavia eroja codec-tuissa. Uusia natiiveja tageja on myös lisätty, sekä html-koodin ulkoasua on pyritty yksinkertaistamaan. Nämä muutokset parantavat saatavuutta eri laitteiden välillä, joten erillisiä web-sovelluksia sisällön näyttämiseen ei tarvita. (Cardinal 2011.)

Web-kehittäjille uusi html-versio tuo paljon uusia mahdollisuuksia sivujen toiminnallisuuteen sekä visuaaliseen ilmeeseen vaikuttamiseen, sekä kehityskustannukset alenevat kun ei tarvitse enää ottaa vanhojen selainten rajoituksia huomioon toteutuksissa. (Wołejko 2012).

2.5 Cascading Style Sheet

CSS eli Cascading Style Sheet –kieli on suosituin kieli XHTML-asiakirjojen tyylisivujen kirjoittamiseen. Sen ovat kehittäneet Bert Bos ja Håkon Lie, jotka ovat mukana W3C:ssä. Tavoitteena oli kehittää tyylisivukieli, jolla olisi helppo kontrolloida ulkoasua ja jolla kokemattomatkin voisivat helposti luoda omia tyylisivujaan. Ajatuksena oli sisällyttää CSS HTML:ään erikoistagien ja –attribuuttien avulla, tai laittaa sisältö omaan erilliseen tiedostoon. Jälkimmäinen menettely sisältää tyylisivun linkittämisen asiakirjaan, joten tyylisivun tiedostotunnus on erilainen. (Boumphrey ym. 2001, 236.)

Nimi CSS eli Cascading Style Sheets tarkoittaa, että asiakirjan lopullista ulkoasua tuottamassa voi toimia vuorovaikutuksessa useampi kuin yksi tyylisivu. Eri lähteistä saatavia yksittäisiä tyylisivuja voidaan yhdistää sekä lyhyitä tyylisivuja voidaan kirjoittaa CSS:n avulla. Tyylisivu korvaa selaimen sisäisen tyylisivun muutamia ominaisuuksia. (Boumphrey ym. 2001, 238.)

Vuonna 1996 julkaistiin alkuperäinen määrittely, CSS 1. Se keskittyi pääasiasssa melko yksinkertaisiin tyylisivutoimintoihin kuten väriin, fonttiin sekä taustakuvaan. CSS 2 julkistettiin vuonna 1998 ja se sisältää joitakin kehittyneempiä piirteitä aiempaan versioon verrattuna. Ne liittyivät sivupohjaiseen layoutiin (tulostusta varten), ladattavien fonttien tukeen sekä suorakaiteenmuotoisten alueiden määrittelyyn asiakirjaosien esittämistä varten, jolloin asiakirjojen layoutin määrittely oli vapaampaa ilman HTML-taulukoita. (Boumphrey ym. 2001, 236.)

CSS 3 eli CSS-tyyliohjekielen seuraava versio, jota on kehitelty jo erittäin pitkään, luetaan joskus myös HTML 5-käsitteen alle. Useimmat CSS 3:n osista ovat vielä keskeneräisiä, mutta monia niissä esitellyistä piirteistä on jo toteutettu selaimiin. (Korpela 2011, 285.)

CSS 3 on laaja kokonaisuus, jonka eri osia on kehitelty eri tahtiin. Toisaalta CSS 3 ei mullista CSS:n teknistä rakennetta, vaan vanhaan perusrakenteeseen ollaan lisäämässä suuri joukko ominaisuuksia. Se on suhteellisen itsenäinen kokonaisuus,

jota voidaan käyttää eri HTML:n versioista huolimatta ja myös muiden merkkauskielten kuten XML:n yhteydessä. (Korpela 2011, 285.)

Yksi yhtymäkohta CSS 3:n ja HTML 5:n välillä on se, että canvas-elementtiin piirrettäessä värien määrittelyt ovat CSS 3:n Color-moduulin mukaiset. Tähän liittyy mahdollisuus käyttää ns. alfa-kanavaa säätämään läpinäyvyyttä. Myös selaimeen latautuvat fontit eli ns. web-fontit saatetaan mainita HTML 5:n yhteydessä. Ne kuitenkin kuuluvat CSS:ään ja fonttitekniikoihin, joskin HTML 5 – luonnoksiin sisältyy jo mahdollisuus käyttää web-fontteja, kun canvas-elementtiin kirjoitetaan tekstiä. (Korpela 2011, 285.)

CSS 3:n uutuuksista jotkin ovat sellaisia, että ne vastaavat HTML 5:n kanssa samoihin tarpeisiin mutta vain eri tekniikoilla. Esimerkiksi CSS 3:ssa tekstin varjostuksen voi tehdä text-shadow-ominaisuudella, mutta myös HTML 5:n canvas-elementtiin liittyvällä SetShadow-funktiolla. (Korpela 2011, 285.)

CSS3:n avulla voidaan tehdä elementtien rajoista pyöreitä, sekä värimaailmaa on parannettu. Uusia värimalleja ovat HSL, HSLA, ja RGBA, ja niiden läpinäkyvyyttä voidaan säätää uudella ”A” –arvolla. Kuville voidaan nykyään määrittää varjostusta, joka oli ennen mahdollista vain kuvanmuokkaus-ohjelmistoilla. (Silva 2010)

3 UUDET OMINAISUUDET

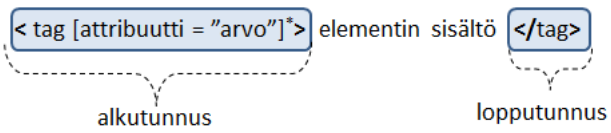
HTML 5:n perusideana on että sitä ei miellettäisi yhtenä kokonaisuutena, vaan pyrkimys on muodostaa pienistä kokonaisuuksista toimiva paketti jolla tehdä uusia innovatiivisiä ratkaisuja. (Flores 2011.) HTML 4:n menestyksen myötä uusi standardiversio ei muuta merkkauskielen perusasioita. Esimerkiksi HTML 4 – standardin mukaan ohjelmoitu web-sovellus tulee toimimaan myös HTML 5-aikana. (Pilgrim 2012.)

Uuden tekniikan avulla web-sisältö on jäsennelty aiempaa semanttisemmin ja rakennetta on pyritty yksinkertaistamaan. Tämän myötä ylimääräiset elementit ja koodi-kutsut eivät ole enää tarpeen samassa mittakaavassa. Uudet ominaisuudet parantavat erityisesti mobiililaitteiden käyttökokemusta, koska vanhanaikaisella tekniikalla toteutetut web-sovellukset ovat aiheuttaneet suorituskyky-ongelmia. (Why HTML5 Rocks 2013)

Semanttisuus tarkoittaa että elementtien tyylit (väri, fontti jne.) määritellään erillisessä CSS-tyylitiedostossa, eli ei käytetä esim. Otsikko. Ideana on, että sivustojen sisältö kootaan eri lähteitä yhdistelemällä. Mikrosemanttisessa webissä ajatusta viedään vielä pidemmälle laittamalla esimerkiksi verkkokaupan tuotteen hinnan <price> -tagiin ja sen ominaisuudet omiin tageihinsa. Tämän hyötynä on esimerkiksi hakukoneet jotka voivat oppia tekemään tuotevertailuja tämän avulla. (Miettinen 2012.)

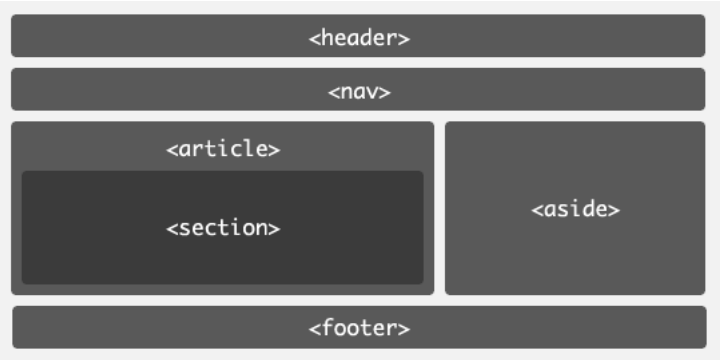
3.1 Elementit

Dokumenttityyppi määritellään HTML 5-mukaisesti <!DOCTYPE>, jonka avulla selain tulkitsee sivun standardisti ja vanhemmissa selaimissa dokumentti tulkitaan HTML 4-dokumentiksi. Aiempien standardiversioiden määrittelyt ovat monimutkaisempia koska niissä vaaditaan tunniste-osia joilla viitataan DTD (document type definition) –määrittelyihin. (Hyrskykari 2012.)



Kuva 2. HTML-elementin syntaksi. (Hyrskykari 2012.)

Kuva 3 havainnollistaa tyypillistä kahden palstan rakennetta jossa on korvattu tyypilliset div-elementit uuden standardin elementeillä:



Kuva 3. HTML 5 dokumentin tyypillinen rakenne. (Hunt 2007.)

header

Header-elementti on koko sivun tai sivun osan ylätunniste. Eli se vastaa tavanomaista sivujen ”yläbanneria”. Se on HTML 5-luonnoksissa hieman epämääräisesti määritelty; se sisältää navigointivälineitä tai johdattavaa sisältöä. Tätä tarkennetaan kuitenkin sanomalla, että se sisältää yleensä otsikon (h1-h6 tms elementin tai hgroup-elementin) ja että siinä voi olla myös hakulomake, sisällysluettelo tai logoja. (Korpela 2011, 94.)

nav

Nav-elementti sisältää linkkejä sivun muihin osiin tai muille sivuille. Se sopii erityisen hyvin sivuston vakionavigoinnin merkkaamiseen. Jos sivun alussa on sivun sisällysluettelo, sen ympärille on luonnollista kirjoittaa nav-merkkaus. Nav-elementti ei itsessään luo linkkejä, vaan sen sisälle kirjoitetaan linkit normaaliin tapaan. (Korpela 2011, 97.)

Nav-elementti on tarkoitettu merkkaamaan lähinnä sivun tärkeimpiä navigointiosia. Esimerkiksi linkkilistaa, joka sisältää aiheeseen liittyviä linkkejä muihin sivustoihin, ei pitäisi kirjoittaa nav-elementin sisälle. Vaan sivun lopussa oleva lyhyt linkkilista, jossa viitataan pääsivuun, käyttöehtoihin ym, on luonteeltaan navigointia. Se voidaan

halutessa kirjoittaa nav-elementiksi, mutta footer-elementti riittää yleensä. (Korpela 2011, 98.)

div tai article

Article tarkoittaa artikkelityyppistä sivun osaa, jonka voisi julkaista erillisenäkin. Se koostuu yleensä blogimerkinnästä tai lehtityyppisen sivun yhdestä artikkelista, pakinasta tai muusta vastaavasta. Article-elementti muodostaa melko itsenäisen kokonaisuuden, joka olisi toiselle sivulle kopioitavissa tai muutenkin eri tarkoituksiin käytettävissä, esimerkiksi jaettavissa eri julkaisijoille jonkin järjestelmän kautta. (Korpela 2011, 86, 90.)

Seuraavassa on yksinkertainen esimerkikki article-elementistä. Article-elementissä on usein otsikko ja tieto kirjoittajasta. Ne eivät ole kuitenkaan pakollisia, ja usein varsinkin blogimerkinnöistä ne saattavat puuttua. (Korpela 2011, 90.)

```
<article>
<p>
<time datetime=”2013-03-27”>27.3.2013</time>
Tänään on keskiviikko. Mukavan aurinkoinen päivä.
</p>
</article>
```

section

Section-elementti sisältää jonkin osan dokumentista, sen avulla voidaan määritellä dokumentin rakenne hierarkisessa järjestyksessä, esimerkiksi:

```
<section>
<h1>Ensimmäisen luvun otsikko</h1>
<!—Sisältöä />
</section>
<section>
<h1>Toisen luvun otsikko</h1>
<!—Sisältöä />
</section>
```

aside

Aside-elementti tarkoittaa jonkinlaista sivuasiaa tai reunahuomautusta, vaikka sen ulkoinen esitysasu voikin vaihdella (W3C 2011). Sen sisältö on jotakin sivun varsinaisen asian yhteydessä mainittavaa, sitä jotenkin sivuavaa ja sellaista, että se voitaisiin myös jättää pois. Tällainen elementti voisi olla konkreettisesti raunahuomautuksena eli marginaalissa olevana sisältönä kirjamuotoisessa tekstissä. (Korpela 2011, 90.)

Käytännössä aside-elementti on yleensä syytä muotoilla jollakin tapaa muusta tekstistä poikkeavaksi. Muulloin lukijan on vaikeaa tai jopa mahdotonta hahmottaa sen sisältöä oikein. Kaksipalstaisen esityksen käyttö on yksinkertainen muotoilukeino. Siinä oikeanpuoleinen palsta sisältää aside-elementit. Tällöin on kuitenkin riskinä, että kyseinen palsta tulkitaan esimerkiksi mainospalstaksi ja jätetään näin huomiotta. Aside-elementistä voidaan tehdä myös kelluva (display: float) tai se se voidaan sijoittaa asemoinnilla tekstin viereen erilliseksi laatikoksi. (Korpela 2011, 91.)

footer

Footer-elementti tarkoittaa alatunnistetta. Se sisältää sivun osan tai koko sivun alatunnisteen. Linkki organisaation pääsivulle, tekijänoikeusmerkintä, päiväys, tieto tekijästä ja linkit aiheeseen liittyviin sivuihin ovat tyypillistä footer-elementin sisältöä. Jos mukana on tekijän yhteystietoja, ne pitäisi kirjoittaa address-elementin sisään. (Korpela 2011, 86, 94.)

Hyvin yksinkertainen esimerkki sivun lopussa olevasta footer-elementistä:

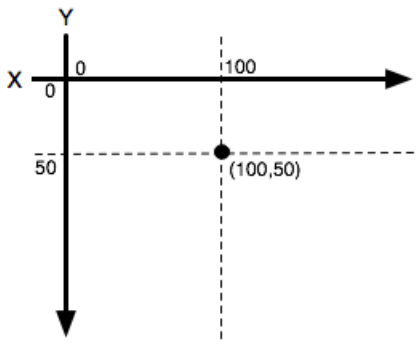
<footer>
© 2013 Satakunnan ammattikorkeakoulu. Sivuston käyttöehdot
</footer>

3.1.1 Canvas

Canvas-elementti on yksi keskeisimmistä uudistuksista HTML 5-standardissa, jolla tehdään piirtoalusta grafiikan luomiseen. Itse grafiikkaa luodaan Canvas 2D –rajapinnalla jota määritellään Javascriptin funktioilla, niillä voidaan luoda lähes kaikenlaista grafiikkaa aina yksinkertaisista viivoista partikkelinäytöksiin. Tällä hetkellä Canvas pystyy esittämään vain 2D-grafiikkaa, tulevaisuuden määrittelyissä myös 3D aiotaan ottaa mukaan. (Sheridan 2011.)

Itse elementti on yksinkertainen, se luo vain suorakulmaisen alueen jossa ei ole sisältöä, sen reunaviivat täytyy määritellä erikseen. Sille määritellään piirtoalueen koko width- ja height-määreillä, ja sille annetaan yksilöllinen id jotta siihen voi viitata myöhemmin helposti. Piirtämiskonteksti luodaaan getContext-metodilla, tämän luoman olion metodeilla voidaan vaikuttaa grafiikan ulkonäköön. (Korpela 2011, 200.)

Itse piirtoalue on kaksiulotteinen ruudukko, johon piirrettävien objektien dimensiot määritellään koordinaatein X/Y akseleilla, joiden origo on vasemmassa yläkulmassa. X-akselilla arvot suurenevat kohti alueen oikeaa reunaa ja Y-akselilla alinta reunaa kohden kuten havainnollistettuna kuvassa 4.



Kuva 4. Piirtoalueen koordinaatit X/Y -akselilla

Seuraavassa esimerkissä näkyy canvas-alue sekä sen sisään piirretty suorakulmio koordinaatein 0,0,150,75. Kuvan lähdekoodi on luettavissa liitteessä 1.

Kuva 5. Suorakulmio Canvas-elementin sisällä.

3.1.2 SVG

SVG (Scalable Vector Graphics) on xml-pohjainen kuvauskieli jolla piirretään web-sivulle 2D-grafiikkaa –sekä sovelluksia. Sitä käytetään usein erilaisten taulukoiden sekä 2D-kaavioiden tekoon X/Y –koordinaattien avulla. Nykypäivänä kaikki suosituimmat selaimet pystyvät esittämään svg-grafiikkaa, jopa IE:n uusimmissa versioissa ei tarvita enää erillistä katselinta svg-grafiikalle. (Tutorialspoint 2013.)

SVG ja Canvas ovat molemmat grafiikan tuottamiseen tarkoitettuja objekteja mutta ne toimivat eri tavoin. Canvas tuottaa bittikartta-grafiikkaa eli kuva koostuu pikseleiden väriarvoista, kun taas SVG tuottaa vektorigrafiikkaa. SVG on siis xml-koodia binäärisen pikselidatan sijaan. Kuva säilyttää täyden tarkkuuden suurennuksissa sekä pienennyksissä näytön erotuskyvyn rajoissa. (Elliot 2013.) Seuraavassa kaaviossa vertaillaan Canvas ja SVG:n hyviä ja huonoja puolia:

	Canvas	**SVG**
Edut	• Tuottaa hyvin suorituskykyistä grafiikkaa • Manipulaatio pikselin tarkkuudella • Piirtoalue voidaan tallentaa kuvatiedostoksi	• Vektoripohjainen • Hyvä tuki animaatioille • Elementtien muokkaus DOM-mallin mukaan

haitat	• Puuttuu rajapinta animaatioille • Ei skaalautuva • Ei sovellu käyttöliittymiin	• Hidastuu lukuisten elementtien kanssa • Ei sovellu pelisovelluksille

Taulukko 1. Canvas ja SVG:n vertailua. (Marakana 2012.)

3.2 Rajapinnat

Ohjelmointirajapinnat eli API:t (Application programming interface) ovat tapa ohjelmoida sovelluksia käyttäen esi-tehtyjä komponentteja, jotka mahdollistavat eri ohjelmien vaihtaa tietoja eli keskustella keskenään. Websivut kuten Twitter ja Youtube julkaisevat omia rajapintojaan jotta web-kehittäjät voivat sisällyttää niiden toimintoja omille sivuilleen. Yksi ohjelmointirajapintojen päätavoitteista on standardoida ja yksinkertaistaa eri mekaniikkojen toimintatapoja jotka muuten olisivat monimutkaisia käyttää.

3.2.1 WebSocket

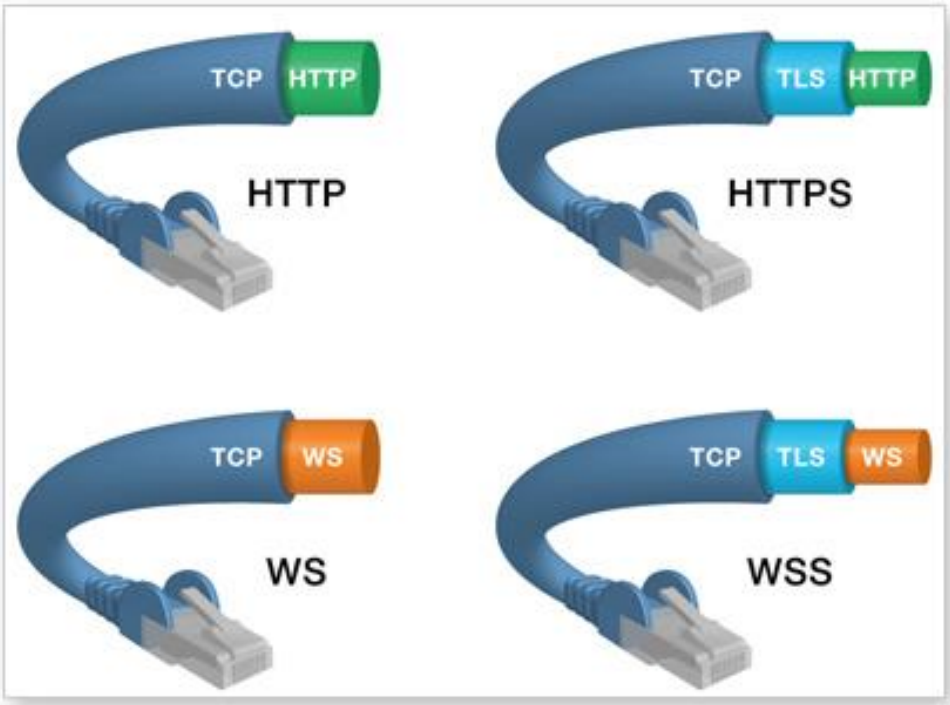
WebSocket API on rajapinta joka on kehitetty kaksisuuntaiseen viestintään sovellusten välillä. Se on erillinen luonnos HTML 5-spesifikaatiosta mutta se mielletään samaan kokonaisuuteen sen teknisen sovellettavuuden myötä. (Korpela 2011, 292.) Teknologioita on ollut jo pitkään saatavilla jotka lähettävät tietoa heti kun sitä on saatavilla palvelimella, mutta niiden käyttämän HTTP:n myötä ne eivät sovellu sovelluksiin jotka vaativat pientä vasteaikaa, Esimerkiksi internetissä pelattaville vuorovaikutteisille peleille. (Ubl & Kitamura 2010.)

Normaalisti kun selain tekee yhteyden web-sivulle, se lähettää HTTP-pyynnön palvelimelle jolla kyseinen sivu on. Saatuaan pyynnön palvelin lähettää takaisin vastauksen, eli tapahtuman mitä selaimen käyttäjä komennollaan halusi. Useimmiten

jos haluaa web-sivuilla realiaikaista tietoa, täytyy manuaalisesti painaa päivitä-painiketta jatkuvasti, joka ei ole käytännöllinen vaihtoehto. (Lubbers & Greco 2013.)

Web-socketin ajatus on avata selaimen ja palvelimen välille yksinkertaistettu yhteys, jonka kautta lähetetään vain kulloinkin tarvittava data, näin säästetään vasteaikaa normaalisti erikseen avattavan yhteyden sekä erinäisten ohjaustietojen sijaan. Huolena on ollut yksinkertaistetun yhteyden tietoturvallisuus, siksi Web-sockets –eli suomeksi vastakkeet on otettu monista selaimista pois käytöstä tai piilotettu. (Korpela 2011, 292.)

Uudessa sovellusliittymässä voidaan käyttää samaa SSL-salausta kuin HTTP-yhteydessä käyttämällä varmenteita. HTTPS:n avulla asiakasohjelma ja palvelin muodostavat turvallisen yhteyden ja vasta sitten aloittavat HTTP-protokollan. Web Socketit eli vastakkeet voivat samalla tavoin muodostaa tietoturvallisen WSS –eli Web Socket Secure-yhteyden. WSS-yhteys toimii samalla salausprotokollalla kuin tietoturvallinen HTTPS-yhteys. (Zimmermann 2012.) Kuvassa 2. havainnollistetaan WSS:n ja HTTP:n salattujen yhteyksien samankaltaisuus.



Kuva 5. Web-vastakkeiden TLS/SSL suojaus. (Zimmermann 2012.)

3.2.2 Web Storage

Web Storage on uusi rajapinta-luonnos joka parantaa perinteisten evästeiden tapaa tallentaa selaimen selaustietoja lokaalisti. Evästeiden avulla voimme kirjautua sivustoille automaattisesti joita käytämme usein. Niiden huonona puolena on pidetty huonoa tietoturvaa koskien haku- ja selaushistoriaa. Toisena huonona puolena on pidetty niiden kapasiteettirajoja jotka ovat useimmissa selaimissa 4 KB per eväste. Siksi web-kehittäjät usein tallentavat käyttäjätietoja myös omille palvelimilleen. (Kappert 2011.)

Web Storage –ohjelmointirajapinnalla voidaan tallentaa arvoja lokaalisti web-selaimessa. Evästeiden tapaan data säilyy käyttäjän sivuilta poistumisen tai selaimen sulkemisen jälkeenkin. (Pilgrim 2011.) Uudella rajapinnalla on mahdollista tallentaa käyttäjätietoja 5-10 MB per verkkotunnus. Tämä mahdollistaa paljon suuremman käyttäjätietomäärän tallennuksen kuin perinteisillä evästeillä, esimerkiksi käyttäjäasetuksia, lokalisointia, tai väliaikaisen offline-tietovaraston. Selain ei myöskään palauta tietoa palvelimelle toisin kuin evästeissä. (Kappert 2011.)

Web Storage –rajapinta on tehty hyvin yksinkertaiseksi, määrittelyssä on vain kaksi oliota: SessionStorage ja LocalStorage. SessionStorage on tarkoitettu vain selaimen välilehti –tai ikkunasessioon, eli se on tarkoitettu vain yhden web-istunnon kerraksi. LocalStoragessa tieto säilyy myös eri selain-istuntojen välillä. Tämä tarkoittaa että data on saatavilla ikkunan/välilehden tai selaimen sulkemisen jälkeenkin. (Kappert 2011.)

Rajapinnan kahdella oliolla on metodit setItem ja getItem, joilla tallennetaan ja haetaan tieto muistista. Rajapinnan määrittelyssä puhutaan avain/arvo –pareista, joihin laitetaan tallennettavan datan nimi ja arvo määritetään merkkijonoina. (Korpela 2011, 232.) Esimerkkinä tallennetaan luku 2013 jolle annetaan avain eli se tallennetaan nimellä ”vuosiluku”:

```
localStorage.setItem(”vuosiluku”, ”2013”;
```

Paikallisessa muistissa oleva tieto vastaavasti saadaan millä tahansa seuraavista lausekkeista:

localStorage.getItem(”vuosiluku”)
localStorage.(”vuosiluku”)
localStorage.vuosiluku

3.2.3 Drag and drop

Drag and drop on tapahtumapohjainen Javascriptillä määriteltävä rajapinta joka mahdollistaa lähes minkä vain elementin liikuttamisen web-sivulla. (Bidelman 2012.) Yksinkertaistettuna Drag and drop-rajapinnassa operaatio aloitetaan hiiren alasklikkaus-tapahtumalla ja siirto-tapahtuman jälkeen ”tiputtaminen” tapahtuu hiirinapin vapauttamis-tapahtumalla. (Hickson & Schwarz 2013.)

Ensin tapahtumakäsittelijän täytyy tarkistaa että vedettävä valinta ei ole tekstiä, sen jälkeen käsittelijä tallentaa datan DataTransfer-olioon ja asettaa sallitut toiminnot eli kopiointi, liittäminen jne. (Hickson & Schwarz 2013.) Vaikka HTML 5:n natiivi rajapinta liikutettaville objekteille on mutkikkaampi ja hankalampi kuin vastaavilla esimerkiksi JQuery-ratkaisuilla, silti natiivisti selaimessa saa tehtyä tehokkaampia ja nopeampia sovelluksia. (Bidelman 2010). Liitteessä 2 on aiheesta yksinkertainen esimerkkikoodi, jossa vedetään suosikkihedelmät toiseen paikkaan.

3.2.4 Geolocation

Geolocation paikannus-rajapinta mahdollistaa käyttäjän sijaintitietojen saannin, myös liikkuvalla laitteella. Rajapinnalla ei ole väliä miten se saa sijaintitiedon, kunhan asiakassovellus voi pyytää ja vastaanottaa tiedon standardilla tavalla. Tekniikka voi olla esimerkiksi GPS tai WiFi. Käyttäjä voi myös syöttää tiedon manuaalisesti. (Mahemoff 2010.) Rajapinnan spesifikaation mukaan on tärkeää käyttäjän

yksityisyyden vuoksi saada ensin lupa paikannukseen ennen sen suorittamista. Selain suorittaa tämän erillisen laatikon avulla tai sivun ylhäällä tai alhaalla selaimesta riippuen, jossa kysytään lupaa käyttäjän paikannukseen. (Devlin 2011.)

Selaimen käyttäjän sijaintitiedot voidaan saada monin eri tavoin, joilla on vaihtelevat tarkkuudet. Pöytäkone-sovellus käyttää todennäköisesti joko WiFI-paikannusta tai IP:n mukaan, joka voi usein antaa vääriä tietoja. Mobiilipaikannuksessa käytetään tekniikoita, kuten GPS- tai WiFI- ja GSM/CDMA -tekniikoita. (Devlin 2011.)

Rajapinnalla on kaksi päämetodia: getCurrentPosition() hakee käyttäjän sen hetkisen sijaintitiedon, ja watchPosition() päivittää liikkuvan laitteen sijaintitietoja. Näiden metodien synnyttämä Position-olio tallentaa tiedot sijainnista, joita ovat koordinaatit sekä tiedot etenemisvauhdista sekä suunnasta. (Tutorialspoint 2013.) Olion kaikki ominaisuudet on selvitettynä seuraavassa taulukossa:

Nimipituus	**Merkitys**	**Tyyppi**	**Yksikkö**
coords.latitude	leveys	double	aste
coords.longnitude	pituus	double	aste
coords.accuracy	tarkkuus	double	metri
coords.altitude	korkeus	double t. null	metri
coords.altitudeAccuracy	korkeuden tarkkuus	double t. null	metri
coords.heading	suunta	double t. null	aste
coords.speed	vauhti	double t. null	m/s
timestamp	paikannushetki	Date t. null	

Taulukko 2. Position-olion ominaisuudet. (Korpela 2011, 243.)

Position-olion ominaisuudet coords.heading ja coords.speed eivät ole yleensä käytössä, sillä niillä kuvataan käyttäjän liikettä mobiili-laitteella. Jos käytössä on aiempia paikannustuloksia, niiden avulla määritellään käyttäjän suunta ja nopeus. Jotta sivu pystyy näyttämään käyttäjän kartalla liikkuvana pisteenä, watchPosition-funktiota kutsutaan sivulla aina kun sijainti muuttuu. (Korpela 2011, 243.)

Esimerkissä käyttäjän sijaintitiedot tulevat koordinaatteina Alert-ikkunaan.

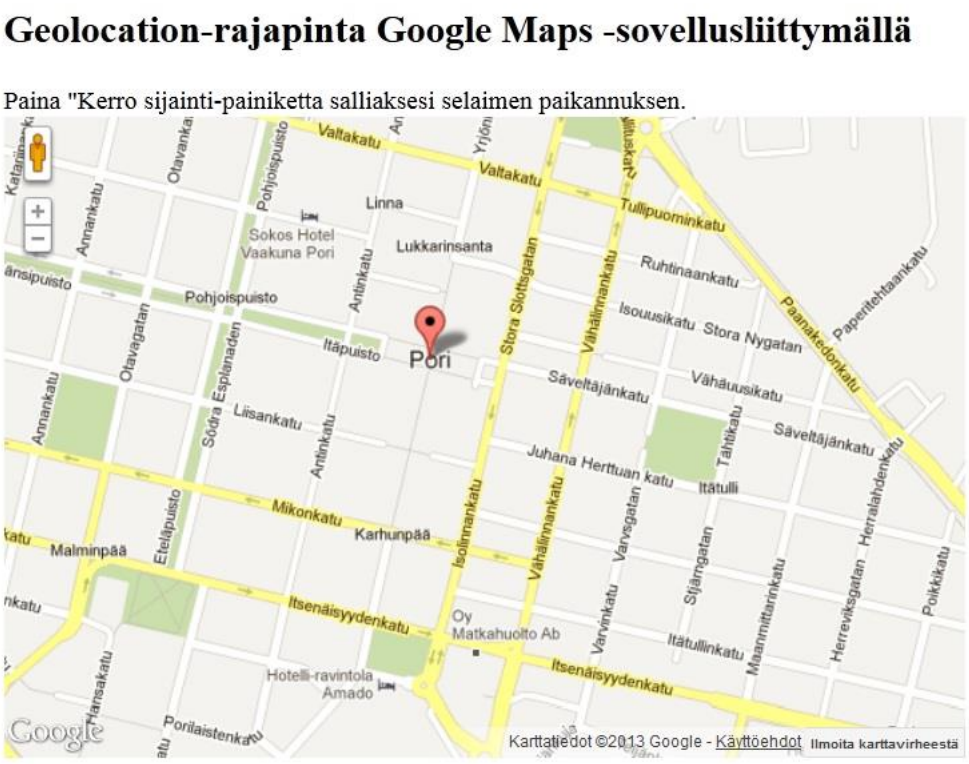
```
if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(
    displayPosition,
    displayError,
    { enableHighAccuracy: true, timeout: timeoutVal, maximumAge: 0 }
  );
}
else {
  alert("Selaimesi ei tue Geolocation-rajapintaa");
}

function sijainti(position) {
  alert("Latitude: " + position.coords.latitude + ", Longitude: " + posi-
tion.coords.longitude);
}
<!--virhekoodien  selitykset -->
function displayError(error) {
  var virheet = {
    1: 'lupa evätty',
    2: 'Paikannus ei saatavissa',
    3: 'Pyyntö aikakatkaistu'
  };
  alert("Error: " + virheet[error.code]);
}
```

Google Maps tarjoaa ns geocoding-rajapintoja, joilla voidaan muuttaa koordinaatteja paikkojen nimiksi ja toisin päin. Rajapinnan saa käyttöön seuraavalla Javascript-määrittelyllä:

```
<script type="text/javascript"
```

```
  src="http://maps.googleapis.com/maps/api/js?sensor=true">
</script>
```

Google Maps –rajapinnan määritys vaatii sensor-parametrin määrityksen, joka ilmaisee käytetäänkö paikannuksessa sensoria, kuten GPS-paikanninta. Parametrin arvo on joko tosi tai epätosi (true/false). (Google Developers 2013) Liitteessä 4 on koodiesimerkki, jolla määritellään oma sijainti Google Maps-rajapinnan avulla. Koodissa Javascriptiä käyttämällä paikannetaan käyttäjän sijainti ja laitetaan punainen merkki sen kohdalle, joka näkyy kuvassa 6.



Kuva 6. Sijainti Google Maps -sovellusliittymässä Geolocation-rajapinnan avulla.

3.3 Multimedia

Videota, ääntä tai muuta media voidaan liittää HTML-dokumenttiin linkillä, esimerkiksi TestiVideo. Linkin painaminen käyttöjärjestelmästä ja selaimesta riippuen käynnistää selainlaajennoksen, joka

kykenee toistamaan ääntä/videota tms. (Korpela 2011, 178.) HTML 5:ssä on natiivi tuki videolle ja audiolle, eli niiden toistamiseen ei tarvita erillistä selainlaajennusta. Tämä mahdollistaa sivuston sisällön toistamisen riippumatta laitteesta tai selaimesta. (Paavilainen 2012.) HTML 5:n mukainen merkintätapa videon toistoon tehdään <video> tagilla, esimerkiksi <video src=”video.webm”>
</video>. Tällä hetkellä selainten välillä on eroja erilaisten videoformaattien tuessa, siksi videolle kannattaa merkitä useampi formaatti, jotta kaikki selaimet pystyvät sitä toistamaan. Selain käy kaikki lähdemerkinnät läpi ja valitsee sen, joka on kyseisessä selaimessa tuettuna:

```
<video>
  <source src="video.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"' />
  <source src="video.webm" type='video/webm; codecs="vp8, vorbis"' />
</video>
```

(Delgado 2010.)

Kun sivulle upotetaan video, siinä näkyy aluksi videon ensimmäinen kuvaruutu. Videon alla on ohjainkontrolleja, joilla voi käynnistää/pysäyttää videon sekä säätää äänenvoimakkuutta. Kuva 8 on otettu natiivista videosta Internet explorer-selaimessa.

Kuva 8. Videon toisto natiivisti selaimessa. (Quirksmode 2013.)

Tällä hetkellä vaikeinta ei ole HTML 5:n mukainen videon upottaminen websivulle vaan eri selainten tukivaihtelu videoformaateissa. Tällä hetkellä on kolme eri

formaattia: H.264, Ogg Theora, ja VP8 (WebM). Taulukko 3 selvittää eri formaattien tuen selaimissa.

	H.264	Ogg Theora	VP8 (WebM)
	native	with install	with installs
	native for now; with install from Microsoft	native	native
	native	with install	no
	with install from Microsoft	native	native
	no	native	native

Taulukko 3. Selainten tuki eri videoformaateille. (Whitney 2013.)

Äänen toisto selaimessa oli ennen mahdollista vain erillisen siirrännäisen avulla, esimerkiksi Adobe Flash. Nykyään HTML 5-standardissa on mahdollista toistaa ääntä natiivisti selaimessa ilman erikseen asennettavia ohjelmia. Merkintä tapahtuu <audio> -tagilla, jota kaikki nykyaikaset selaimet nyt tukevat. Elementille on valittavissa seuraavat attribuutit:

src	URL-osoite jonka polkuna on lähdetiedosto
autoplay	totuusarvo joka määrittää tiedoston automaattisen toistamisen
loop	totuusarvo joka määrittää tiedoston jatkuvan uudelleentoiston
controls	määrittää selaimessa näytettävät median kontrollit
preload	määrittää tiedoston esiladattavaksi selaimeen

Taulukko 3. Video-elementin saatavissa olevat attribuutit (Boas 2009.)

Koska eri selaimet tukevat eri audion codec-formaatteja, sivun lähdekoodiin kannattaa laittaa useampi formaatti lähdetiedostolle, myös flash-formaattina. Esimerkkinä kolme yleisintä formaattia:

```
<audio controls preload="auto >
  <source src="testi.mp3" />
```

```
 <source src="testi.ogg" />
<source src="testi.waw" />
 <!-- Flash fallback-->
</audio>
```

(Boas 2009.)

4 SELAINTUKI

Verkkosivujen esittämisessä olevat erot selainten välillä riippuvat ennen kaikkea siitä, millainen taittokomponentti niissä on. Taittokomponentti (layout engine) tarkoittaa selaimen osaa, joka muotoilee HTML-dokumentin nähtävissä olevaksi esitykseksi. Se käyttää tietynlaista esitystapaa, joka saattaa olla eri selainten taittokomponenteissa erilainen. Se ottaa myös sivujen CSS-tyyliohjeet huomioon, joka myös vaihtelee eri selainten CSS-tuissa. Usein komponentti perustuu ohjelmakoodiin, jota voidaan käyttää erilaisissa selaimissa sekä myös muun muassa sähköposti-ohjelmissa, jotka osaavat näyttää HTML-muotoisia viestejä. (Korpela 2011, 33.) Yleisimmät selaimet käyttävät seuraavia taittokomponentteja: Trident (Internet explorer), Gecko (Firefox), Webkit (Chrome, Safari), Presto (Opera). (Elayidom 2008.)

Selainten HTML 5-tukea voidaan testata esimerkiksi Javascript-ehdoilla. Esimerkkinä ehto if (”localStorage” in window). Tämä testaa, tunteeko selain oliota window.localStorage, joka viittaa web-muistiin. HTML 5:n uudet ominaisuudet on myös usein määritelty niin, että ne voivat tarjota vaihtoehtoisen sisällön kyseisen ominaisuuden tuen puuttuessa selaimessa. (Korpela 2011, 33.)

Esimerkkinä:

```
 <canvas id="CanvasEsim" width="200" height="100">
  <-- Vaihtoehtoinen sisältö: -->
Canvas ei ole tuettu selaimessasi.
</canvas>
```

Hyvä keino selvittää selaimen tuki HTML 5:lle on mennä sivulle http://html5test.com. Sivu pisteyttää selaimen tuen eri ominaisuuksille 0-500 pisteeseen. Liite 4 selvittää tämän hetkisen tuen uusille ominaisuuksille neljässä suosituimmassa selaimessa. Testin mukaan Google Chrome olisi tällä hetkellä kirkkaasti uutta standardia tukevin selain, ja Internet Explorer olisi vähiten yhteensopiva uusille ominaisuuksille. Testissä yllätti Operan ja Mozilla Firefoxin tulos, sillä pisteytyksessä Opera oli Firefoxia edellä mm. uusien lomakeominaisuuksien tuen myötä. Testin tulos saattaa silti hämätä, sillä Firefox on edelleen Chromen kanssa edelläkävijä uusien rajapintojen ja tietovarasto-ominaisuuksien tuen myötä.

5 YHTEENVETO

Uuden HTML-standardin kehitystyö on kestänyt jo vuosia, ja tulee kestämään vielä pitkälle tulevaisuuteen. Työn edetessä huomasin kuinka laaja uusi spesifikaatio on, siksi rajasin aihe-alueen vain tärkeimpiin uudistuksiin sekä uusiin rajapintoihin, jotka mielletään HTML 5-spesifikaatioon. Työtä tehdessä muodostui kuitenkin hyvä kokonaiskäsitys standardista ja sen myötä uusista trendeistä web-kehityksessä.

Vaikka kehitystyö on vielä kesken, suosituimmissa selaimissa on jo tuki uusille elementeille ja rajapinnoille Internet Exploreria lukuunottamatta. Etenkin mobiilipuolella HTML 5 on jo laajasti käytössä uusien kosketusominaisuuksien myötä. Ominaisuuksien tuen puuttumisen varalle on kuitenkin fallback-varatoimintoja, sekä uusi standardi on suunniteltu taaksepäin yhteensopivaksi vanhemmille selainversioille. Koodin ulkoasua on muutettu onnistuneesti helppolukuisemmaksi ja semanttisemmaksi. Uuden CSS 3 –spesifikaation avulla saadaan helpommin ja tehokkaammin sekä toimivia että näyttäviä web-sovelluksia.

Mobiililaitteilla käytetään runsaasti enemmän natiiveja sovelluksia kuin perinteisiä web-sivustoja. Uuteen HTML-standardiin on sisällytetty runsaasti mobiilikäyttöä

helpottavia ominaisuuksia, jotta saataisi vaihtoehtoisia tapoja kehittää mobiilipalveluita. The Guardianin (Berliner, 2013) artikkelin mukaan luotto perinteisiin mobiilisovelluksiin on yhä korkea, koska HTML 5:en työkalut eivät vielä ole vaadittavalla tasolla. Tämä tulee vielä todennäköisesti muuttumaan, sillä web-sovelluksiin siirtymällä ei tarvitse maksaa provisioita mobiilialustan sovelluskaupalle.

Natiivin videon toisto kannustaa siirtymistä HTML 5:een varsinkin mobiilisovelluksissa, koska se on varteenotettava vaihtoehto Adobe Flashille. Esimerkiksi Apple rajoittaa jyrkästi Flashin käyttöä mobiililaitteissaan, täten kannustaen HTML 5:en käyttöä mobiilisovelluksissa. Vielä on tosin epävarmaa, kuinka paljon natiivia videon toistoa tullaan hyödyntämään toteutuksissa, koska selainkehittäjillä on vielä ristiriitoja videoformaateissa. Pidän todennäköisenä, että HTML 5 tulee syrjäyttämään Adobe Flash:in käytön videotoistossa täysin, mutta tämä vaatii isoimpien Flashia tällä hetkellä käyttävien yritysten siirtymistä natiivin videon toistoon, joka voi viedä vielä vuosia.

Tulevaisuudessa natiivista 3D-grafiikan esittämisestä tulee suuri edistysaskel webille; erilaisia grafiikkaesityksiä ja pelejä voidaan toteuttaa suoraan selaimessa ilman erillisiä lisäosia. Kehitteillä on monia erilaisia tekniikoita, muun muassa CSS animation, 3D Canvas sekä WebGL (html5rocks 2013). Viime vuosien web-trendeistä päätellen HTML 5 tulee varmasti lyömään läpi mobiililaitteilla, mutta muuten siitä ollaan vielä paikoin skeptisiä. Monet ominaisuudet ovat vielä keskeneräisiä ja hankala käyttää, aika näyttää kuinka hyvin niitä tullaan käyttämään hyödyksi.

LÄHTEET

Bellis, M. 2013.The History of HTML. Viitattu 20.3.2013. http://inventors.about.com/od/computersoftware/a/html.htm

Berners-Lee, T. 1991. X11 BROWSER for WWW. Viitattu 21.3.2013. http://lists.w3.org/Archives/Public/www-talk/1991SepOct/0003.html

Bidelman, E. 2012. Native HTML5 Drag and Drop. Viitattu 23.3.2013. http://www.html5rocks.com/en/tutorials/dnd/basics/

Boas, M. 2009. Native Audio in the browser. Viitattu 15.4.2013. http://html5doctor.com/native-audio-in-the-browser/

Boumphrey, F., Greer, C., Ragget, D., Ragget, J., Schnitzenbaumer, S. & Wugofski, T. 2001. Inside XHTML: Ohjelmoijan käsikirja. Helsinki: Edita.

Bright, P. 2012. W3C announces plan to deliver HTML 5 by 2014, HTML 5.1 in 2016. Viitattu 22.3.2013. http://arstechnica.com/information-technology/2012/09/w3c-announces-plan-to-deliver-html-5-by-2014-html-5-1-in-2016/

Cardinal, D. 2011. What is HTML5, and why it will save the web from itself. Viitattu 29.4.2013. http://www.extremetech.com/computing/98137-what-is-html5-one

Cox, P. 2011. Top 10 reasons to use HTML 5 right now. Viitattu 29.4.2013. http://tympanus.net/codrops/2011/11/24/top-10-reasons-to-use-html5-right-now/

DeBolt, V. 2009. HTML 5: The nav element. Viitattu 27.3.2013. http://www.webteacher.ws/2009/07/09/html5-the-nav-element/

Delgado, E. 2010. HTML5 Video. Viitattu 12.4.2013. http://www.html5rocks.com/en/tutorials/video/basics/

Devlin, I. 2011. Finding your position with Geolocation. Viitattu 25.3.2013. http://html5doctor.com/finding-your-position-with-geolocation/

Elayidom, J. 2008. A quick look at browser engines. Viitattu 16.4.2013. http://getjins.wordpress.com/2008/09/10/a-quick-look-at-browser-engines-trident-gecko-webkit-presto/

Elliot, I. 2013. Getting started with SVG for HTML5. Viitattu 10.4.2013. http://www.i-programmer.info/programming/graphics-and-imaging/2063-getting-started-with-svg-html5.html

Flores, B. 2011. HTML5 & CSS3: Take Your Design to Another Level. Viitattu 21.3.2013. http://webdesignledger.com/tips/html5-css3-take-your-design-to-another-level

Github. 2012. The HTML5 test. Viitattu 16.4.2013. http://html5test.com/

Google Developers. 2013. Google Maps API. Viitattu 26.3.2013. https://developers.google.com/maps/articles/geolocation

Hickson & Schwarz. 2013. HTML: The Living Standard. Viitattu 23.3.2013. http://developers.whatwg.org/dnd.html#dnd

Hunt, L. 2007. A Preview of HTML 5. Viitattu 27.3.2013. http://alistapart.com/article/previewofhtml5

Hyrskykari, A. 2012. HTML 5 -dokumentti. Viitattu 27.3.2013. http://www.cs.uta.fi/jwt/12/luennot/3-html.html

Jaakkola, T. 2011. Haastaako HTML5 perinteiset käyttöjärjestelmät sovellusalustana? Viitattu 22.3.2013. http://www.gofore.com/blogi/201108/haastaako-html5-perinteiset-k%C3%A4ytt%C3%B6j%C3%A4rjestelm%C3%A4t-sovellusalustana

Kappert, L. 2011. Introduction to HTML5 Web Storage. Viitatttu 22.3.2013. http://sixrevisions.com/html/introduction-web-storage/

Korpela, J. 2011. HTML: uudet ominaisuudet. Porvoo: WSOYpro OY.

Lubbers, G. 2013. HTML5 Web Sockets: A Quantum Leap in Scalability for the Web. Viitattu 22.3.2013. http://www.websocket.org/quantum.html

Mahemoff, M. 2010. A Simple Trip Meter using the Geolocation API. Viitattu 25.3.2013. http://www.html5rocks.com/en/tutorials/geolocation/trip_meter/

Marakana. 2012. Canvas Vs. SVG. Viitattu 10.4.2013. http://marakana.com/bookshelf/html5_tutorial/canvas.html

Miettinen, O. 2012. Mikä ihmeen HTML5? Viitattu 16.4.2013. http://www.myyverkossa.fi/2012/08/mika-ihmeen-html5.html

Nykänen, O. 2003. W3C pähkinänkuoressa. Viitattu 22.3.2013. http://www.w3c.tut.fi/reports/2003/0113aboutw3c/index.html

Paavilainen, J. 2012. HTML5 – mitä kaikkien tarvitsee tietää. Viitattu 12.4.2013. http://www.snoobi.fi/blogi/html5-mita-kaikkien-tarvitsee-tietaa/

Pilgrim, M. 2011. Dive into html 5. Viitattu 21.3.2013. http://diveintohtml5.info/introduction.html

Pilgrim, M. 2011. Introducing HTML5 StorageIntroducing HTML5 Storage. Viitattu 22.3.2013. http://diveintohtml5.info/storage.html

Quirksmode. 2013. How to Embed Video Using HTML5. Viitattu 12.4.2013. http://www.htmlgoodies.com/html5/how-to-embed-video-using-html5.html

Shannon, R. 2012. The History Of HTML. Viitattu 21.3.2013.
http://www.yourhtmlsource.com/starthere/historyofhtml.html

Sheridan, M. 2011. A Developer's Guide to HTML5 Canvas. Viitattu 8.4.2013.
http://www.sitepoint.com/a-developer%E2%80%99s-guide-to-html5-canvas/

Silva, M. 2010. What's New In CSS3? Viitattu 8.4.2013.
http://notesofgenius.com/new-css3/

Tutorialspoint 2013. Geolocation getCurrentPosition() API. Viitattu 25.3.2013.
http://www.tutorialspoint.com/html5/geolocation_getcurrentposition.htm

Tutorialspoint. 2013. HTML5 - SVG Tutorial. Viitattu 10.4.2013.
http://www.tutorialspoint.com/html5/html5_svg.htm

Ubl & Kitamura. 2011. Introducing WebSockets. Viitattu 21.3.2013.
http://www.html5rocks.com/en/tutorials/websockets/basics/

W3C. 2012. Facts About W3C. Viitattu 22.3.2013.
http://www.w3.org/Consortium/facts

W3C. 2009. Frequently Asked Questions (FAQ) about the future of XHTML.
Viitattu 22.3.2013. http://www.w3.org/2009/06/xhtml-faq.html

W3C. 2011. HTML/Elements/Aside. Viitattu 27.3.2013.
http://www.w3.org/wiki/HTML/Elements/aside

W3C. 2012. HTML 5. Viitattu 27.3.2013.
http://www.w3.org/TR/html5/sections.html#the-article-element

WHATWG. 2013. FAQ. Viitattu 22.3.2013. http://wiki.whatwg.org/wiki/FAQ

Whitney, J. 2013. How to Embed Video Using HTML5. Viitattu 12.4.2013.
http://www.htmlgoodies.com/html5/how-to-embed-video-using-html5.html

Wilson, B. HTML 2.0. Viitattu 21.3.2013.
http://www.blooberry.com/indexdot/history/html20.htm

Wolejko, G. 2012. 5 reasons Why HTML5 matters? Viitattu 29.4.2013.
http://www.cognifide.com/blogs/ux/5-reasons-why-html5-matters/

World Wide Web Foundation. 2008. Sir Tim Berners-Lee. Viitattu 21.3.2013.
http://www.webfoundation.org/about/sir-tim-berners-lee/

Zimmermann, R. 2012 HTML5 WebSocket Security is Strong. Viitattu 22.3.2013.
http://blog.kaazing.com/2012/02/28/html5-websocket-security-is-strong/

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<canvas id="CanvasEsim" width="200" height="100" style="border:1px
solid #c3c3c3;">
</canvas>

<script>

var c=document.getElementById("CanvasEsim");
var malli=c.getContext("2d");
malli.fillStyle="#FF01324";
malli.fillRect(0,0,150,75);

</script>
</body>
</html>
```

```
<p>Minkälaisista hedelmistä pidät?</p>
<ol ondragstart="dragStartHandler(event)">
 <li draggable="true" data-value="hedelma-omena">Omenat</li>
 <li draggable="true" data-value="hedelma-paaryna">Päärynät</li>
 <li draggable="true" data-value="hedelma-appelsiini">Appelsiinit</li>
</ol>
<script>
  var internalDNDType = 'text/x-example';
  function dragStartHandler(event) {
    if (event.target instanceof HTMLLIElement) {
      event.dataTransfer.setData(internalDNDType,
event.target.dataset.value);
      event.dataTransfer.effectAllowed = 'move';
    } else {
      event.preventDefault();
    }
  }
</script>
```

```
<p>Tiputa lempi-hedelmäsi tänne:</p>
<ol dropzone="move string:text/x-example" ondrop="dropHandler(event)">
</ol>
<script>
 var internalDNDType = 'text/x-example';  function dropHandler(event) {
   var li = document.createElement('li');
   var data = event.dataTransfer.getData(internalDNDType);
   if (data == 'hedelma-omena') {
     li.textContent = 'Omenat';
   } else if (data == 'hedelma-paaryna') {
     li.textContent = 'Päärynät';
   } else if (data == 'hedelma-appelsiinit') {
     li.textContent = 'Appelsiinit';
   } else {
     li.textContent = 'Tuntematon hedelmä';
   }
   event.target.appendChild(li);
 }
</script>
```

```
<script>
var coords = new google.maps.LatLng(position.coords.latitude, position.coords.longitude);
  var options = {
    zoom: 15,
    center: coords,
    mapTypeControl: false,
    navigationControlOptions: {
    style: google.maps.NavigationControlStyle.SMALL
    },
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };
  var map = new google.maps.Map(document.getElementById("mapcontainer"), options);
  var marker = new google.maps.Marker({
      position: coords,
      map: map,
  });
</script>
```

	Internet Explorer 10	Opera 12.15	Mozilla Firefox 19	Google Chrome 19
Jäsennyssäännöt tokenizer tree building SVG MathML	0/10	10/10	0/10	10/10
Canvas canvas element 2D context Text	20/20	20/20	20/20	20/20
Video video element Subtitle support MPEG-4 support H.264 support Ogg Theora support WebM support	21/30	30/30	21/30	30/30
Audio PCM audio support AAC support MP3 support Ogg Vorbis support Ogg Opus support WebM support	20/20	20/20	20/20	20/20
Elementit New or modified elements Global attributes or methods	15/35	25/35	26/35	30/35
Lomakkeet Field types Fields Forms	7/115	115/115	61/115	110/115
Käyttäjä-integrointi Drag and drop HTML editing Spellcheck	18/20	19/20	20/20	20/20
History and navigation Session history	0/10	10/10	10/10	10/10
Microdata Microdata	0/15	15/15	15/15	0/15
Web-sovellukset Application Cache Custom scheme handlers Custom content handlers	1/20	19/20	20/20	18/20

Turvallisuus Sandboxed iframe Seamless iframe iframe with inline contents	0/20	0/20	10/20	20/20
Sekalaiset Scoped style element Asyncronous script execution Runtime script error reporting	1/10	2/10	5/10	5/10
Sijainti ja suunta Geolocation Device Orientation	15/20	15/20	20/20	20/20
WebGL 3D context	0/25	10/25	25/25	25/25
Kommunikointi Cross-document messaging Server-Sent Events WebSocket	5/35	33/35	35/35	35/35
Tiedostot File API	0/10	10/10	10/10	10/10
Storage Session Storage Local Storage IndexedDB	10/25	15/25	25/25	25/25
Workers Web Workers Shared Workers	0/15	15/15	10/15	15/15
paikallinen multimedia Access the webcam	0/10	10/10	10/10	10/10
Ilmoitukset Web Notifications	0/10	0/10	0/10	10/10
Muut Page Visibility Text selection Scroll into view Mutation Observer	5/10	7/10	10/10	10/10
Audio Web Audio API	0/5	0/5	0/5	5/5
Video ja animaatio Full screen support Pointer Lock support	0/10	4/10	10/10	10/10
	Internet Explorer 10	**Opera 12**	**Mozilla Firefox 19**	**Google Chrome 19**
Yhteensä:	138	404	393	468

Taulukko 4. Uusien ominaisuuksien tuki eri selaimissa. (Github 2012.)